ARDUINO

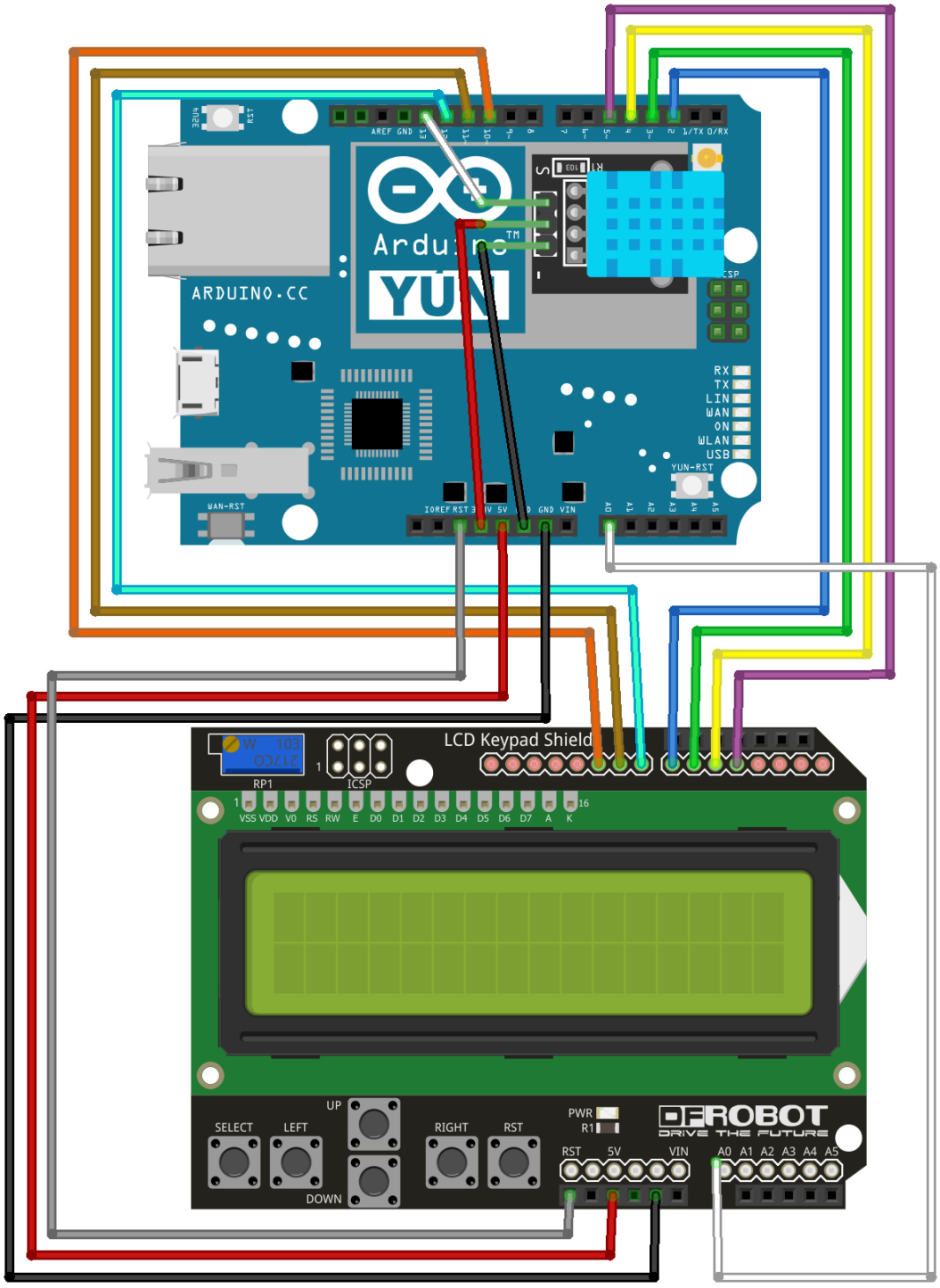
Proyecto dispositivos ubicuos – David Flaity Pardo

Indice

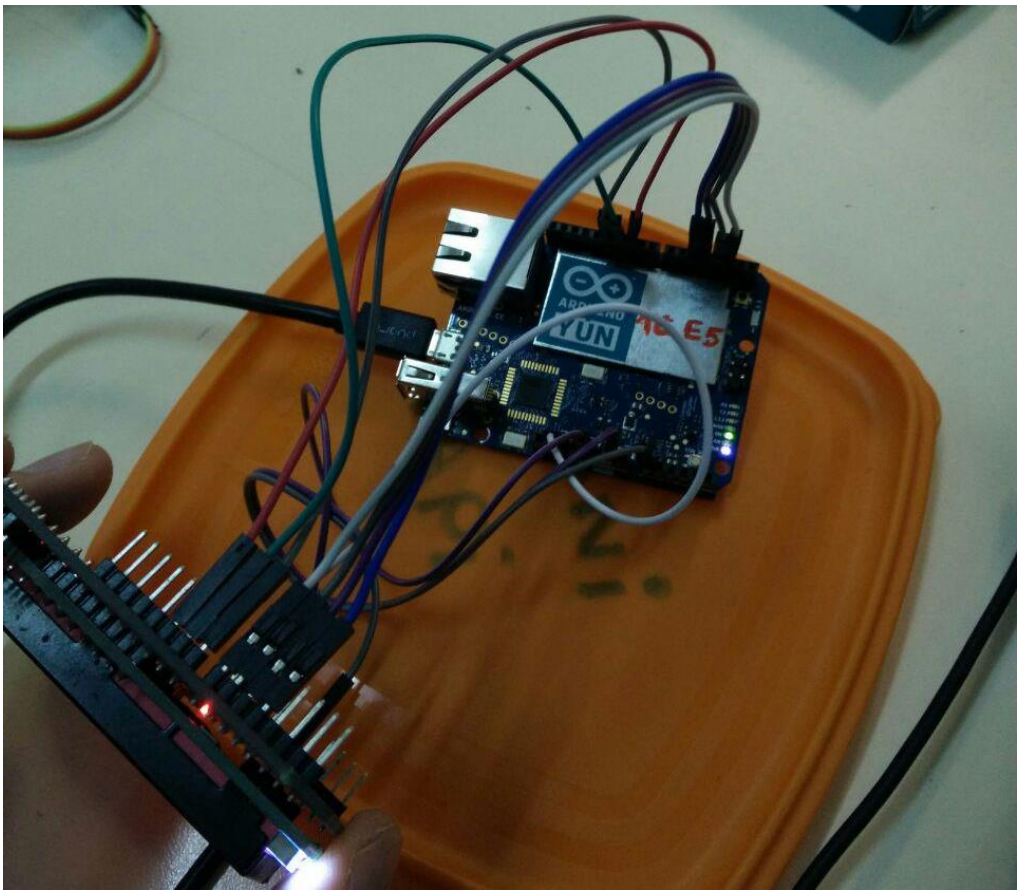
Diagrama de montaje

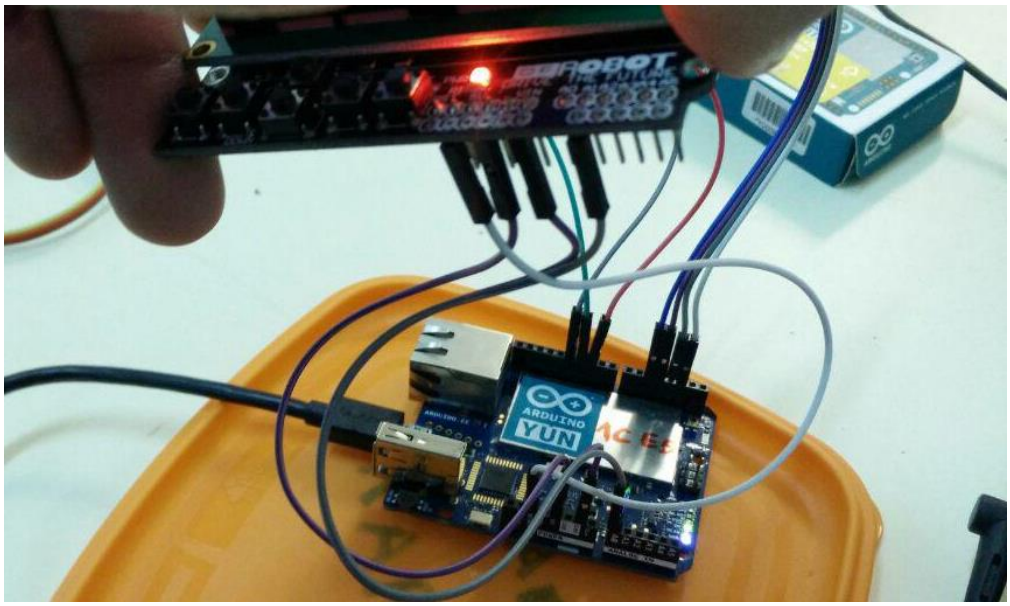
Esquema en fritzing

Los colores de los cables son todos distintos, para facilitar el entendimiento del esquema. No coinciden con los colores de los cables de la imagen real.



Fotos reales del montaje





Funcionalidades requeridas

.- Monitorización de valores de humedad y temperatura cada 0.1 segundos.

.- Mostrar dichos valores en la pantalla LCD.

.- Guardar un valor de lectura cada 30 segundos en una base de datos.

.- Cuando se superen ciertos umbrales, hacer parpadear la pantalla.

Monitorización valores y escritura en pantalla LCD

```
   #include "dht.h"
   #include <LiquidCrystal.h>
   #define DHT11_PIN 9
   dht DHT;
   LiquidCrystal lcd(RS, EN, D4, D5, D6, D7);

   float temperature = DHT.temperature;
   float humidity = DHT.humidity;
   char t[6]=""; // Buffer big enough for 5-character float
   char h[6]=""; // Buffer big enough for 5-character float
   dtostrf(temperature, 3, 1, t);
   dtostrf(humidity, 3, 1, h);
   char temp[16]="Temperature "; // Buffer big enough for 5-character float temperature 49
   char hum[16]="Humidity ";
   strcat(temp, t);
   strcat(hum, h);
   writeInLcd(temp, hum);

   void writeInLcd(char* firstLine, char* secondLine){
      lcd.clear();
      lcd.print(firstLine);
      lcd.setCursor(0, 1);
      lcd.print(secondLine);
}
```

Guardado de valores en Base de datos

Se ha intentado realizar con una librearía, pero siempre que se subía el sketch se corrompía el bootloader. En el proyecto se realizará así. Se añade el código de la librería.

```
#ifndef Sqlite_h
#define Sqlite_h
#ifdef DEBUG
 #define DEBUG_PRINT(x)  Serial.print (x)
#else
 #define DEBUG_PRINT(x)
#endif
#include <Process.h>

class Sqlite{
  public:
    Sqlite(char* dbPath, int led_built_in);
    void sendCommand(String command);
  private:
    char path;
};
#endif
```

SQLITE.H

```
#include "Sqlite.h"

Sqlite::Sqlite(char* dbPath, int led_built_in){
  pinMode(led_built_in, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);
  path = dbPath;
  if(!Serial){
    Serial.begin(9600);
    while (!Serial);
  }
  Bridge.begin();
  digitalWrite(LED_BUILTIN, LOW);
}

void Sqlite::sendCommand(String command){
   Process p;
   String cmd = "sqlite3 -line ";
   String dbCommand = String(" '"+command+"'");
   p.runShellCommand(cmd+String(path)+dbCommand);
   while (p.available()>0) {
     char c = p.read();
     DEBUG_PRINT(c);
   }
}
```

SQLITE.cpp

Umbrales

```
#define highTemp 40
#define lowTemp 0
#define highHum 80
#define lowHum 20

 if ((humidity < lowHum) || (humidity > highHum) || (temperature < lowTemp) ||
(temperature > highTemp))
     blinkScreen();

void blinkScreen(){
  for(int i=0; i<10; i++){
    lcd.noDisplay();
    delay(readingDelay);
    lcd.display();
    delay(readingDelay);
  }
}
```

TROUBLESHOOTING

Primero, he intentado que funcione con la librería, pero siempre corrompía más tarde.

Luego he decidido a probar realizando un método directamente en el sketch, pero nunca creaba la base de datos ni daba más respuesta. Si intentaba ver que me devolvía la consola de Linux mediante Process.available(), me devolvía basura sin parar. De forma infinita.

Así que el sketch no funciona. Se resolverán sus problemas cuanto antes.